Slide 1

Administrivia

- Homework 3 on Web; due next week.

Slide 2

Variables in C — One More Thing

- Unlike Java, C explicitly defines unsigned types. Useful and appropriate for, e.g., indices.

Slide 3

Arrays in C — Example

- Let's write a sort program . . .
- (Where to get input? let's just generate random values, using library function `rand()`.)

Slide 4

Strings in C

- Java has a `String` class with many useful features and methods. In C that's not possible . . .
- Instead, in C, strings are arrays of `char`s, with the convention that the actual text of interest is followed by a null character (8-bit zero, represented in code as `'\0'`.
- You can operate on individual characters however you see fit; there are also standard library functions for some common operations (e.g., `strcmp` to compare two strings — similar to `compareTo` in Java).
- A significant source of potential trouble — most functions assume that strings are properly terminated, and (worse) many have no safety check to make sure you don't overflow a destination array.

Slide 5

Pointers in C

- Pointers in C are similar to, but not identical to, references in Java — with the key differences having to do with safety features and level of abstraction. (No surprise!)
- In C, pointers are just memory addresses — but they are declared to point to variables (or data) of a particular type. Example:

```
int * pointer_to_int;
double * pointer_to_double;
```

Slide 6

Pointers in C — Operators

- & gets the address of something in memory. So for example you could write

```
int x;
int * x_ptr = &x;
```

- * "dereferences" a pointer. So for example you could change X above by writing

```
*x_ptr = 10;
```

- You can also perform arithmetic on pointers (e.g., `++x_ptr`) — something not allowed in Java, and another example of the languages' different design goals.

Slide 7

Pass By Reference (Sort Of)

- A significant potential limitation on functions is that a function can only return a single value. Pointers provide a way to get around this restriction: By passing a pointer to something, rather than the thing itself, we can in effect have a function return multiple things.
- To make this work, typically you declare the function's parameters as pointers, and pass addresses of variables rather than variables.
- The "sort of" of the title means that this isn't true pass by reference, as it exists in some other languages such as C++, but it can be used to more or less get the same effect. Notice also that Java can't do this, though again there are mechanisms that can more or less get the same effect. (What?)

Slide 8

Pointers Versus Arrays

- In C, pointers and arrays are in some sense(s) equivalent — not identical, but in many contexts interchangeable.
- This is reflected in the `man` pages for many functions (e.g., `printf`). It also means that when you pass an array to a function, what you're actually passing is a pointer — so the array is not copied.

Slide 9

Minute Essay

- None — sign in.